A Appendix

A.1 Interaction Design

SemiConLens provides a set of interactions to support the design tasks. The interactions are designed to be consistent across different views, enabling users to seamlessly transition between views and maintain a coherent mental model for the sequential exploration. The interactions are as follows:

Lasso. Users can draw a lasso selection in the Discovery View to select multiple compounds for cluster and reference filtering.

Brushing. Users can use brush in the Attribute Distribution of the Filter View to define value ranges for range filtering.

Clicking. Users can click on a glyph in the Discovery View to access detailed compound information in the Comparison View. Additionally, clicking on a filter state in the Exploration History navigates users to trace back and forth through the filtering process.

Highlighting. Users can hover over a glyph in the Comparison View to highlight the corresponding compound in the Discovery View and Filter View. Additionally, hovering over a filter state in the Exploration History highlights the corresponding attribute distribution in the Attribute Distribution.

Tooltip. *SemiConLens* provides tooltips in the Discovery View and Comparison View to display detailed attribute information upon hovering over a specific glyph or heatmap cell. The tooltip includes the exact attribute value and uncertainty.

A.2 Data sources and preprocessing

Several major databases provide material attributes. The C2DB[1] reports computational properties of 16,905 2D materials, the Materials Project[2] covers 169,385 compounds, 2DMatPedia[3] compiles 6,351 entries from theory and experiment, and eTran2D[4] focuses on transport data for 91 compounds.

We used C2DB as the primary source due to its size and consistent DFT framework, merging it with mobility values from eTran2D. The combined data were stored in MySQL and filtered (0.2- 4.0 eV band gap, dynamic stability, real phonons, non-magnetic). This yielded 1,339 candidates, with mobility values for 50 compounds and GW-calculated gap, CBM, VBM for 225.

[1] https://2dhub.org/c2db/c2db.html
[2] https://next-gen.materialsproject.org/
[3] http://www.2dmatpedia.org/
[4] https://sites.utexas.edu/yuanyue-liu/etran2d/

A.3 The System Workflow

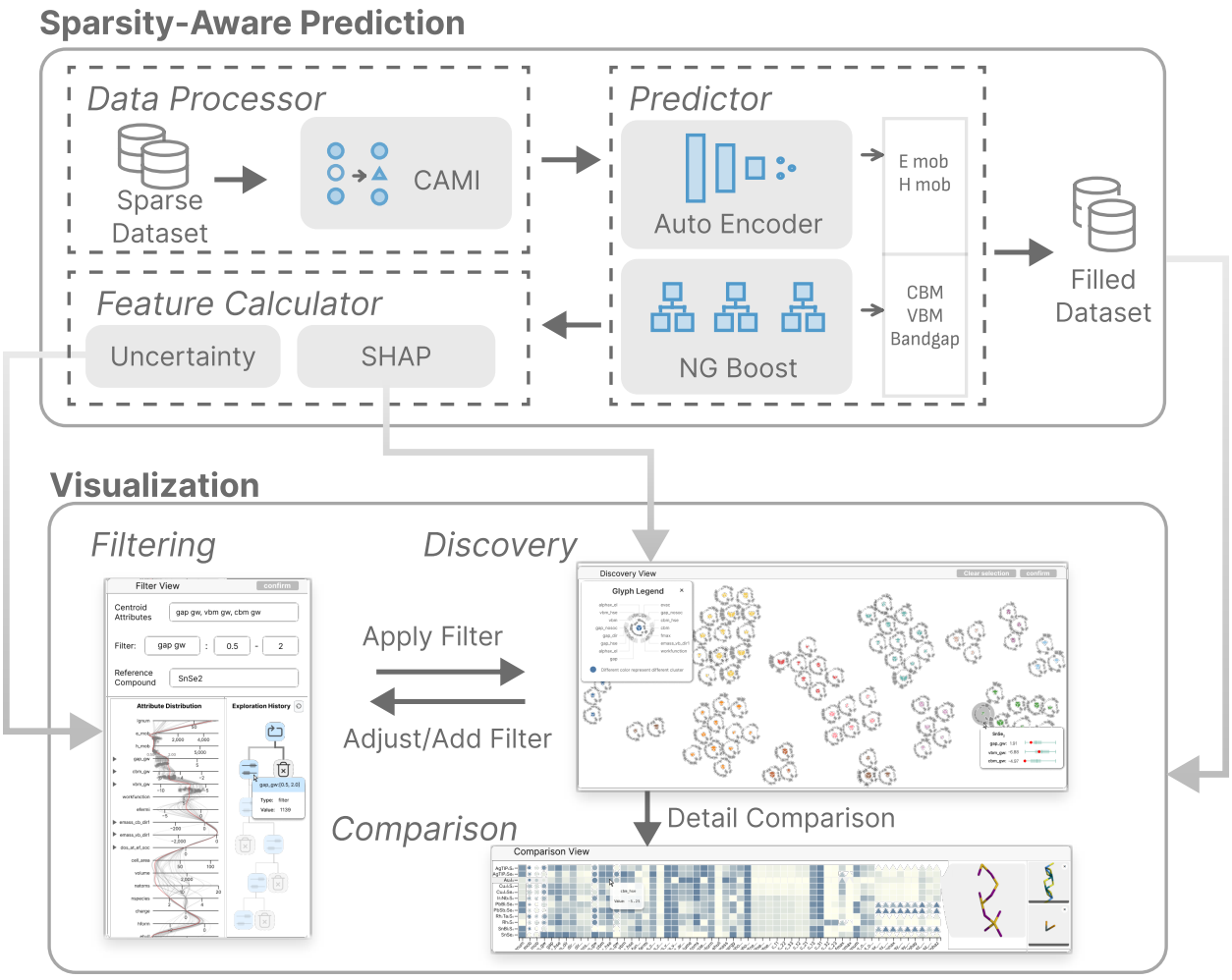

Fig. 1. The workflow consists of two main components: (1) a sparsity-aware prediction pipeline that processes sparse datasets through CAMI data processing, followed by prediction using Auto Encoder and NG Boost models, with uncertainty and SHAP-based feature calculation for interpretability; (2) a visual analysis system, which enables interactive *filtering*, *exploration* and *comparison* of the filled dataset.

A.4 AE Finetuning

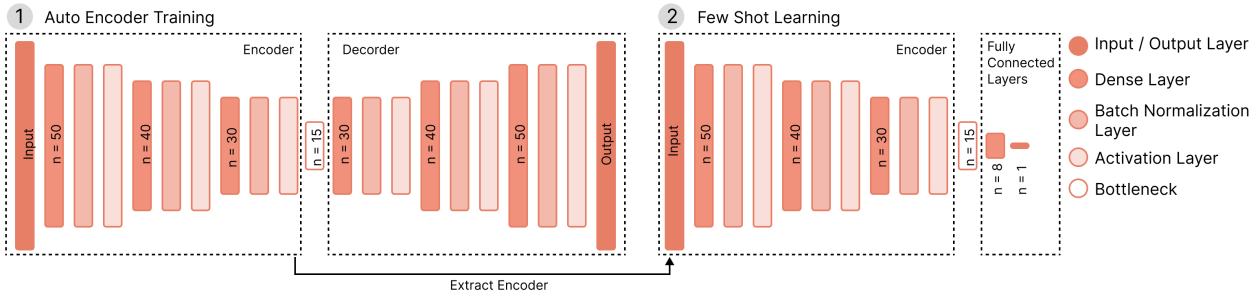

Fig. 2. Configuration of the Auto Encoder and few shot learning model.

As discussed in Quantitative Evaluation in section 7.1, Fig. 2 demonstrates the architecture of the AE in the training and few shot learning. In step Fig. 2 1, an AE was trained on all 1,339 compounds. Then the econder and the bottleneck (latent space) was extracted and the decoder was replaced with a small fully connected network as depicted in Fig. 2 2. For AE training the total number of epochs trained were 1000 with an early stopping mechanism of patience value of 10 on validation loss. Similar for AE few shot learning the patience value was 10 and number of epochs was 200.

A.5 Questions

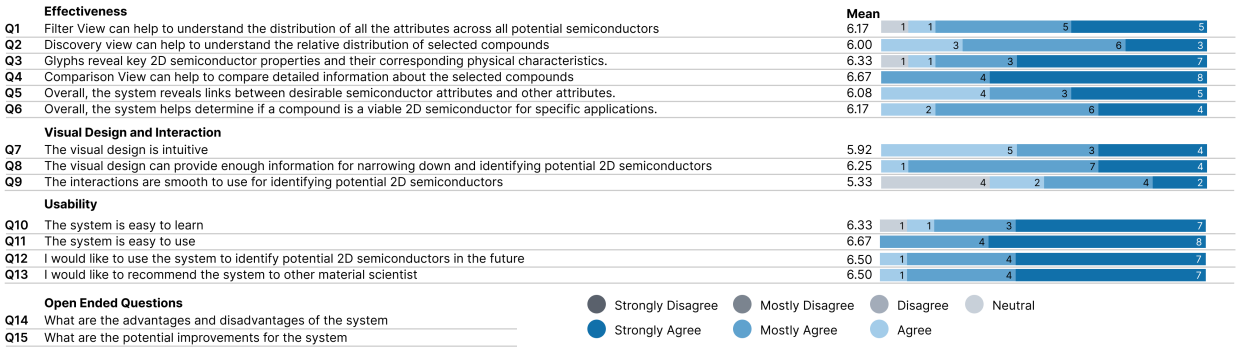

Fig. 3. The expert interview questionnaire results. Q1 to Q13 are close-ended questions which were answered on a 7-point Likert scale. Q14 and Q15 are open-ended questions.

Fig.3 reports the unbridged post study questions we used for the study with the summary of responses from the experts.

A.6 Pseudo code for the Proposed Correlation Aware Multivariate Imputation

Algorithm 1 Correlation Aware Multivariate Imputation (CAMI)

Input: Dataset $X \in \mathbb{R}^{m \times n}$, thresholds A and B (in percentages), and parameters C (number of correlated columns to consider) and D (number of similar rows to consider).

$X \leftarrow X_{:,\mathcal{C}}$ where $\mathcal{C} = \{j \in \{1, \ldots, n\} \mid \frac{|\{i | X_{ij} = \text{null}\}|}{m} \leq A\%\}$
$X \leftarrow X_{\mathcal{R},:}$ where $\mathcal{R} = \{i \in \{1, \ldots, m\} \mid \frac{|\{j | X_{ij} = \text{null}\}|}{|\mathcal{C}|} \leq B\%\}$
$c_m \leftarrow \text{Corr}(X)$
for each column Y in X **do**
    Let j be the index of Y.
    $K(Y) \leftarrow$ the set of top C indices from $\{1, \ldots, n\} \setminus \{j\}$ based on the correlation values in c_m.
    for each $k \in K(Y)$ **do**
        Compute $\mu_k = \frac{1}{|\mathcal{I}_k|} \sum_{i \in \mathcal{I}_k} X_{ik}$, where $\mathcal{I}_k = \{i \mid X_{ik} \neq \text{null}\}$.
        Replace each missing value in column k with μ_k.
    end for
    Let $X' = X_{:,\{j\} \cup K(Y)}$.
    Define $\mathcal{L} = \{i \mid Y_i = \text{null}\}$ and $\mathcal{M} = \{i \mid Y_i \neq \text{null}\}$.
    for each row $P \in \mathcal{L}$ **do**
        Compute the similarity between P and every row $Q \in \mathcal{M}$ using features in $K(Y)$.
        Let $\mathcal{O}_P$ be the top D rows in $\mathcal{M}$ that are most similar to P.
        Impute the missing value in P by $Y_P \leftarrow \frac{1}{D} \sum_{Q \in \mathcal{O}_P} Y_Q$.
    end for
end for

Output: X

Hyperparameters were iteratively tuned for the dataset used. Hyperparameter values were A=40%, B=40%, C=10 and D=10.

A.7 An illustration of the proposed Correlation Aware Multivariate Imputation

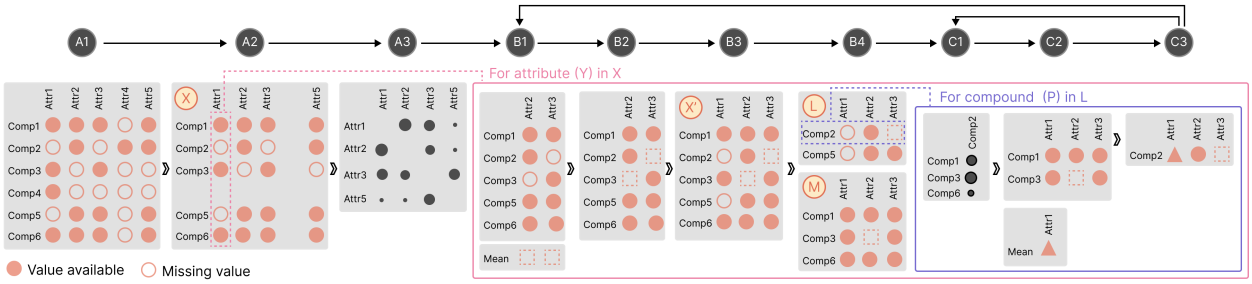

Fig. 4. An illustration of the proposed Correlation Aware Multivariate Imputation. Step A1 : Load the dataset. Step A2 : Select only columns and rows with less than A% and B% missing values respectively. Step A3 : Calculate the correlation matrix. Step B1 : For each attribute (Y) in the filtered dataset X, select top C most correlated columns using the correlation matrix. Then calculate the mean for each selected column. Step B2 : Fill the missing values of each column with the corresponding calculated mean. Step B3 : Create X', a subset of X, using the Y and its most correlated columns mentioned above. Step B4 : split X' into L and M, where column Y is missing and not missing respectively. Step C1: For each compound (P) in L, compute similarity matrix with compounds in M. Step C2: Select the top D most similar compounds from the subset and calculate the mean for column Y. C3 : Fill the missing value of P with the mean calculated above.

A.8 System Interface for *SemiConLens*

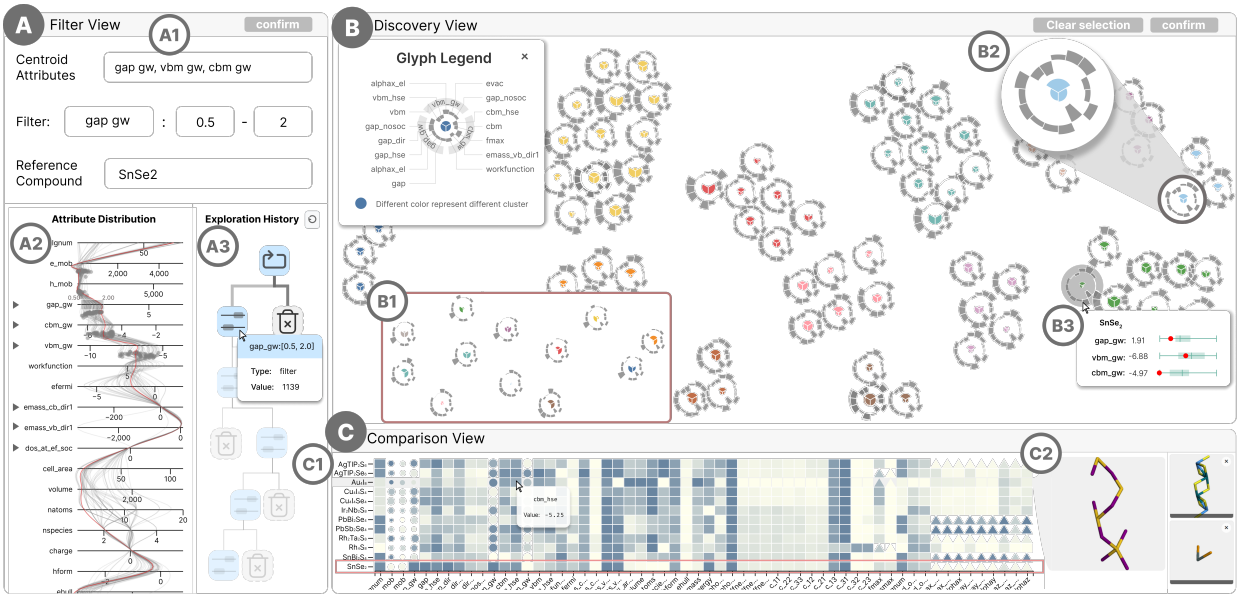

Fig. 5. System interface for *SemiConLens*. Filter View (A) provides an overview of the compound space, applies range-based filters, and tracks exploration history for a more targeted and efficient analysis. Discovery View (B) visualizes clusters and influencing factors of the key attributes, enabling users to discover patterns in the compound space. Comparison View (C) facilitates the comparison of compounds across all attributes and displays their structures.

A.9 Case 1 Exploring clusters - Identifying 2D Materials for Water Splitting

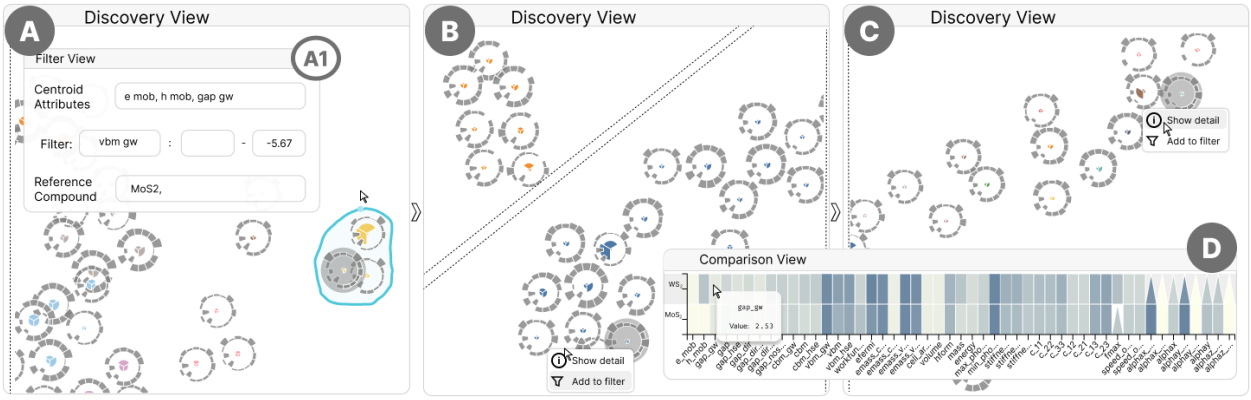

Fig. 6. With *SemiConLens* an expert has identified WS_2 as a similar material to MoS_2, which is suitable for hydrogen evolution reaction in water splitting. First, the expert applied 2 filters, CBM GW > -4.44eV and VBM GW < -5.67 eV using the Filter View. Then they selected MoS_2 as the reference compound as shown in (A1). From the resulting Discovery View, they identified the cluster with MoS_2 and selected it by lassoing the cluster as depicted in (A). They identified the MoS_2 from the resulting Discovery View (B) and add it as the filter for next step. Then they identified WS_2 as the nearest neighbor for MoS_2 from the Discovery View (C) and added both to comparison view for further exploration as shown in (D). Some compounds are excluded from the Discovery View, as indicated by the dotted lines.

A.10 Space Optimization for Cluster-Aware Overlap Removal

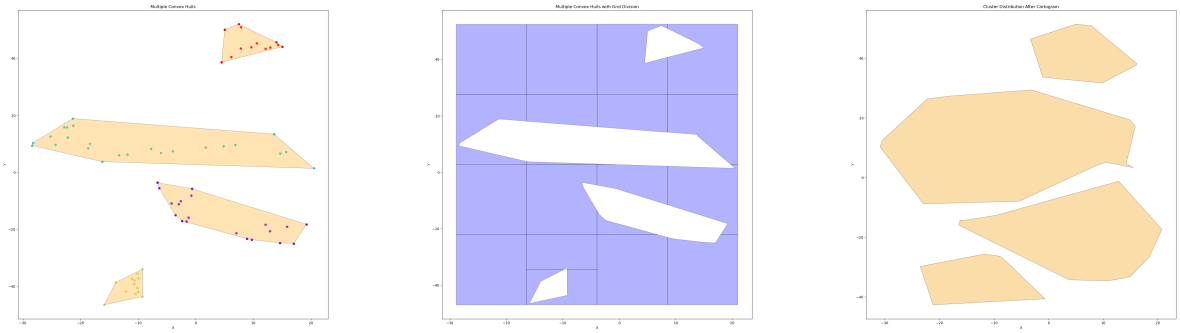

Fig. 7. Examples of intermediate representations in the space-optimization pipeline: (a) cluster polygons constructed from 2D point embeddings; (b) dummy regions generated to fill the remaining space; and (c) the final deformed layout after cartogram transformation.

Cluster Region Construction. The first step transforms discrete point clusters into continuous spatial regions while preserving cluster relationships. Specifically, each cluster is represented as a convex hull polygon P_i, computed using the Quickhull algorithm on the 2D embedding space. For clusters with fewer than three points, we instead generate a circular buffer of fixed radius around the centroid to avoid degenerate polygons.

Dummy Region Generation. Because clustering algorithms often produce non-adjacent cluster polygons, dummy polygons are introduced to fill the empty spaces for subsequent cartogram processing. These dummy regions are generated through a kd-tree inspired recursive partitioning procedure. We first compute a global bounding box covering all cluster polygons and initialize it as a 4×4 grid.

Recursive Subdivision and Clipping. For each grid cell that overlaps the remaining empty space, if the cell fully contains a cluster polygon, we recursively split it along its longer axis until its area falls below 0.1 in normalized units. The resulting regions are then clipped to the empty space, balancing computational efficiency and geometric fidelity.

Weight Assignment and Cartogram Transformation. With this continuous layout in place, we then employ the diffusion-based cartogram technique to optimize polygon areas. Each cluster polygon is assigned a weight proportional to its number of member compounds, while dummy polygons are assigned a small constant weight of 0.1 so that they act only as spacers. This ensures sufficient room for expansion while approximately preserving relative topological relationships.

Pseudo-code.

Algorithm 2 Cluster-Aware Space Optimization

Require: Points $P = \{(x_i, y_i, c_i)\}$ in 2D embedding space
Ensure: Deformed polygons G

Group points by cluster label
for each cluster c_i **do**
    if $|P_i| < 3$ **then**
        Construct a circular buffer around the cluster centroid
    else
        Compute the convex hull polygon P_i using Quickhull
    end if
end for
Compute the global bounding box B covering all cluster polygons
Compute the remaining empty space $R \leftarrow B \setminus \bigcup_i P_i$
Initialize a 4×4 grid over B
for each grid cell g **do**
    if g intersects R **then**
        if g fully contains a cluster polygon **then**
            Recursively subdivide g along its longer axis
            Stop subdivision when cell area < 0.1
        end if
        Clip the resulting cell(s) to R and add them as dummy polygons
    end if
end for
Assign cluster weights according to cluster sizes
Assign dummy polygons a small constant weight of 0.1
$G_0 \leftarrow$ cluster polygons $\cup$ dummy polygons
$G \leftarrow \text{Cartogram}(G_0, w)$
return G